LOW COST PROGRAM GIVES VZ200/300 FULL LEVEL II BASIC

Review of Extended Basic
Tape $18
Disk $25

Ever wished that your little VZ200 or VZ300 would run full Microsoft Level II BASIC instead of just a stripped-down version? You needn't wish any longer thanks to an enterprising local programmer.

Jim Rowe

REMEMBER STEVE OLNEY? If you're a VZ200 or VZ300 owner and BASIC programmer, you should. We've published at least three of his articles so far, mainly on resurrecting dormant functions and statement keywords in VZ BASIC. One was in the March '84 issue, another in October '84 and the last in May '85.

Steve's a very knowledgeable guy when it comes to the VZ200/300, in terms of both software and hardware. He's spent quite a lot of time burrowing into its little secrets, and probably knows as much about it as anyone in Australia.

I know that sounds a bit like paeaning in his pocket, but I've just been trying out the latest fruit of his labours. And this time it's not just an article showing you how to restore a few more missing functions to VZ BASIC. It's a machine language utility program that restores pretty well the whole blinking lot for you — instant Level II BASIC! Hence my little paean of praise.

Steve calls his new utility Extended BASIC Version 2.2, or 'EXBSV2.2' for short. It is available on either cassette tape or disk, to suit both basic and expanded VZ systems. It is also compatible with both the VZ200 and VZ300, and with the current Disk BASIC (V1.2 DOS).

You load EXBSV2.2 into your VZ before you load in anything else. It is only about 1600 bytes long (about 1.5K) and is fully self-locating, finding the top of available RAM and installing itself there. At the same time it lowers the BASIC 'top of RAM' pointer to prevent any other programs from being loaded over it.

As part of the installation it patches itself into ROM BASIC, in much the same way that Disk BASIC does, to become transparent to the user. All that you're aware of is that the RAM is now about 1.5K smaller than before — plus, of course, the fact that your trusty VZ now responds to no less than 25 new BASIC commands!

Of these 25 new commands, 23 are basically resurrected Level II commands that have been sleeping there all the time in the VZ's ROM, quietly waiting for EXBSV2.2 to sound the trumpet. They're listed in the table. The other two are extras — a bonus that Steve Olney has thrown in for good measure. And very handy thay are too: MERGE, to allow you to combine programs and routines, and RENUM to let you rationalise and tidy up a program whose line numbers have become a mess after a lot of editing and patching (or after using MERGE).

All of the 25 new commands are fully functional, and when used in a program can be LISTed — at least on any machine with EXBSV2.2 loaded. All but two of them will even RUN on a VZ which doesn't have EXBSV2.2 loaded! The two exceptions are ON and ERROR, which arise because of a conflict in token codes (normal VZs use the normal ERROR token for the added command SOUND).

Even here Steve Olney has provided an answer, for those who really do want the Level II programs they generate to be capable of running on plain-vanilla VZs (how helpful can the guy get?). He's done this by providing the listing of a short BASIC routine which you can MERGE into the top of your programs after they're finished and debugged. You then use it to convert your finished programs

When it has finished, you DELETE the routine itself (notice that?) and CSAVE the converted program. It won't LIST properly any more, but it will now RUN on a VZ without EXBSV2.2 installed. There's just one tiny catch: you can't use the construct 'IF <expression> THEN ERROR <n>' in any program that you want to convert in this fashion. You can only use ERROR in the 'ON ERROR GOTO' construct. Not a serious limitation, but worth remembering.

But back to EXBSV2.2 itself. Normally you'd expect to load this into your VZ every time you turn it on, which is easy enough and only takes a couple of seconds with the disk system. And with the utility installed, all of the new commands are at your disposal.

It's great to be able to use direct commands like DELETE, AUTO, TRON and TROFF, RENUM and MERGE. How did we ever get along without DELETE? It's so damn useful — not to say virtually essential when you want to scrub a whole range of program lines.

Then into the actual programming. It's really good to be able to use double-precision constants and variables again. Plus to be able to define variables as integer, single, double or string type using DEFINT, DEFSNG, DEFDBL and DEFSTR. It's also much neater to be able to use ON-GOTO and ON-GOSUB, instead of a flock of IF-THENs. Not to mention being able to use ERROR, ERR and ERL. It's nice to be able to use RESUME and RANDOM, too.

Of course there's also FIX, FRE, and MEM — plus familiar old mates like CINT, CSNG and CDBL, POS and STRING$ (handy in setting out screens, that one — I missed it). And of course the very versatile VARPTR. Wheee! Makes

March 84 -
84 - RTTY
85 - video interface ✓
Nov 85 - extended basic ✓

Manual Contents $24.95

CONTENTS

CHAPTER 1 - INTRODUCTION TO MACHINE CODE - Page 1.

1.01 What is Machine Code ? p1
1.02 Digital Voltage Levels p2
1.03 Two Levels - Binary Numbers p2
1.04 Binary Data in the Computer p3
1.05 Machine Code is Complex p3
1.06 Introducing Hexadecimal p4
1.07 Hexadecimal Codes p4
1.08 Mnemonics and Assembly Language p4
1.09 What is an Editor Assembler ? p5
1.10 The Basic Loader p5

CHAPTER 2 - PARTS OF YOUR VZ COMPUTER - Page 7.

2.01 Plastic Chips p7
2.02 Data and Address Busses p7
2.03 Memory Chips p8
2.04 The Microprocessor Chip p8
2.05 Clock Timing p8
2.06 Address Space p9
2.07 Work Space p9
2.08 Pages in Memory p9
2.09 Traffic Along the Data Bus p10

CHAPTER 3 - MACHINE CODE and Hexadecimal Numbers - Page 11.

3.01 Hand Assembling p11
3.02 Hexadecimal Numbers p12
3.03 Binary to Hexadecimal p14
3.04 Decimal to Hexadecimal p14

CHAPTER 4 - LOADING MACHINE CODE PEEK, POKE & USR - Page 17.

4.01 PEEK and POKE p17
4.02 Places Not to POKE p18
4.03 USR Function p19
4.04 Using PEEK to Evaluate System Pointers p19

CHAPTER 5 - THE Z80 REGISTERS - Page 21.

5.01 The Z80 Microprocessor Unit (MPU) p21
5.02 Internal Registers p21
5.03 The Accumulator p21
5.04 The Flag Register p22
5.05 Other Register Pairs p22
5.06 The I and R Registers p23
5.07 The 16-Bit or Address Registers p23
5.08 The Index Registers p23
5.09 The Stack Pointer p23
5.10 The Program Counter p24
5.11 The Alternate Register Set p25

CHAPTER 6 - HARDWARE - CPU, RAM, ROM & Video - Page 27.

6.01 The CPU Busses p27
6.02 Other CPU Inputs p28
6.03 Clock Input p28
6.04 Reset Input p29
6.05 INT Input p29
6.06 Memory Devices p29
6.07 RAM Memory p30
6.08 Static and Dynamic RAM p30
6.09 ROM Memory p31
6.10 The 6847 Video Display Generator p32

CHAPTER 7 - BASIC & MACHINE CODE - Peaceful Co-existence - Page 33.

7.01 Combining Machine Code with Basic p33
7.02 Safe Places for Machine Code p34
7.03 So You Think You're a Programmer ? p35
7.04 VZ Memory Map p35
7.05 Basic ROM p35
7.06 ROM Cartridges p36
7.07 Memory-Mapped I/O p36
7.08 Video Screen RAM p36
7.09 Basic Scratch Pad RAM p37
7.10 Homes for M/C Programs p37
7.11 Embedding Machine Code in Basic p38
7.12 Disadvantages of 'Data Statement' Method p42
7.13 Machine Code Embedded in 'REM' Statements p44
7.14 M/C Between the End of Basic Program and its Variable Storage Area p47
7.15 Passing Values Between Basic and Machine Code Routines p49
7.16 Loading From Disc p50

CHAPTER 8 - ARITHMETIC OPERATIONS - Page 53.

8.01 Numbering Systems p53
8.02 Binary Numbers p54
8.03 Eight-Bit Arithmetic p55
8.04 Adding Two Numbers p56
8.05 Adding Larger Numbers p61
8.06 Manipulating Binary Numbers p64
8.07 Signed Integers p65
8.08 Strings (of Characters) p67

CHAPTER 9 - THE VIDEO SCREEN - Messages & Simple Graphics - Page 69.

9.01 End of Message Flag p69
9.02 Finding the End of Message Flag p69
9.03 Space for Small Test Programs p70
9.04 The Message Program p70
9.05 The Basic Loader p72
9.06 Graphics Program p73

CHAPTER 10 - JUMPS, BRANCHES - (And More on Stack Operations) - Page 77.

10.01 Jumps, Branches and Subroutines p77
10.02 Conditional Tests p78
10.03 Relative Branching p78
10.04 Calculating the Displacement Byte p79
10.05 Two's Complement Numbers p80
10.06 Saving Data on the Stack p81

CHAPTER 11 - EDITOR ASSEMBLER - Page 85

11.01 Definitions p85
11.02 Assembly Language Syntax p85
11.03 Labels p86
11.04 Op-Codes p86
11.05 Pseudo-Ops p86
11.06 Operands p86
11.07 Typical Editing/Assembling Session p89

CHAPTER 12 - PROGRAMMING TECHNIQUES - Page 95.

12.01 Subroutines p95
12.02 Loops p96
12.03 Flowcharts p96
12.04 Example Program p97
12.05 Program Parameters p98
12.06 Assembly Language Source Code Listing p99

CHAPTER 13 - THE Z-80 INSTRUCTION SET - Page 101.

13.01 Mnemonics p101
13.02 Accumulator Operations p103
13.03 Load Instructions p106
13.04 Jumps p108
13.05 Testing p110
13.06 Set and Reset p110
13.07 Rotate and Shift p111
13.08 Increment and Decrement p112
13.09 Input/Output Instructions p112
13.10 Stack Operations p113
13.11 Other Instructions p114

CHAPTER 14 - INPUT AND OUTPUT - The Real World Outside - Page 115.

14.01 Standard I/O p115
14.02 The Printer I/O Port p116
14.03 Memory-Mapped I/O p117
14.04 Cassette Input p119

CHAPTER 15 - CONCLUSION - Where To Go From Here - Page 121.

APPENDICES

APPENDIX 1 - Memory Maps ..... Page 123
- VZ200 ..... p123
- VZ300 ..... p124
- Basic Work Space For 8K VZ200 ..... p125

APPENDIX 2 - Z80 Registers ..... Page 126

APPENDIX 3 - Z80 Pinout ..... Page 127

APPENDIX 4 - Keyboard Layout ..... Page 128

APPENDIX 5 - System Pointers ..... Page 129

APPENDIX 6 - Common Z-80 Opcodes ..... Page 131

APPENDIX 7 - Hexadecimal/Decimal Tables ..... Page 138

APPENDIX 8 - Z-80 Mnemonics Recognised By The VZ Editor Assembler ..... Page 139
- Pseudo-ops Recognised By The VZ Editor Assembler ..... Page 140

you feel a bit like Uncle Scrooge let loose in the Mint (well almost).

All of the new commands and functions seem to work perfectly. I certainly couldn't find any bugs, anyway — if there are any, they're pretty well hidden. From a functional point of view, my VZ now behaves like any other Level II machine.

So thanks to EXBSV2.2, Steve Olney's little genie, you can now trundle out all those old TRS80/System80 programs and get them running on your trusty VZ. The graphics will need a few mods, of course, but the programs themselves will be fine.

And the cost of this magic ute? A mere $15 for the tape version, or $22 for the disk version. Both prices include packing and postage, and EXBSV2.2 comes complete with a set of driving instructions. You couldn't get much better value for money — obviously Steve Olney is not out to rip anyone off.

I've only got one complaint. Couldn't he have given it a name that's easier to pronounce and type, like 'Jeannie'? Try typing EXBSV2.2 all the way through a review, and you'll know what I mean!

Still, whatever he cares to call it, it's a utility that almost every VZ programmer is going to want. And at this stage you can only get it direct from Steve Olney at ~~20th Terrace Parade~~ North Richmond, NSW 2754. I only hope that his local post office is prepared for the onslaught. ●

TABLE 1. WHAT EXTENDED BASIC PROVIDES

System Commands:	
AUTO	automatic line numbering for program entry
DELETE	delete a line or group of lines
TRON	enable trace function (for debugging)
TROFF	disable trace function
MERGE	merge tape program with program in memory
RENUM	renumber program lines
BASIC Statements:	
DEFINT	define variable as an integer
DEFSNG	define variable as single precision
DEFDBL	define variable as double precision
DEFSTR	define variable as string type
ERR	error code
ERL	line in which error was deleted
ERROR	used to simulate an error condition
ON-GOTO	branch to one of several line numbers depending upon the value of an expression
ON-GOSUB	branch to one of several subroutines depending upon the value of an expression
RANDOM	reseed random number generator
RESUME	continue program execution after error handling
BASIC Functions:	
CINT	convert variable to an integer
CSNG	convert variable to single precision
CDBL	convert variable to double precision
FIX	return truncated integer part of a number
FRE	returns the amount of free memory remaining
MEM	returns the amount of free memory remaining
POS	returns the current screen cursor position
STRING$	returns a string of specified length
VARPTR	locates a variable in memory

WRITE TO:

S. OLNEY
P.O. BOX 125
NORTH RICHMOND
NSW 2754.

NOTE:
E.T.I. PUBLISHED THE WRONG BOX NUMBER (125 IS THE CORRECT NUMBER)

REPRINTED BY KIND PERMISSION OF ELECTRONICS TODAY INTERNATIONAL